VirusFinder (version 2)

USER'S MANUAL

VANDERBILT

Bioinformatics and Systems Medicine Laboratory
Vanderbilt University Medical Center
Nashville, Tennessee, USA

January, 2015

TABLE OF CONTENTS

Page No.

1. INTRODUCTION

VirusFinder 2 is a software tool for characterizing intra-host viruses through next generation sequencing (NGS) data. Specifically, it detects virus infection, co-infection with multiple viruses, virus integration sites in host genomes, as well as mutations in the virus genomes. It also facilitates virus discovery by reporting novel contigs, long sequences assembled from short reads that map neither to the host genome nor to the genomes of known viruses. VirusFinder 2 can not only work with the human genome data, but also other organisms with available reference genome sequences (e.g. animals). VirusFinder 2 works with both paired-end and single-end data, unlike the previous 1.x versions that accepted only paired-end reads. The types of NGS data that VirusFinder 2 can deal with include whole genome sequencing (WGS), whole transcriptome sequencing (RNA-Seq), targeted sequencing data such as whole exome sequencing (WES) and ultra-deep amplicon sequencing.

Figure 1 below shows the flowchart of VirusFinder 2, which overall follows a four-step procedure: (1) read subtraction, (2) virus detection, (3) virus integration site detection, and (4) viral mutation detection. By default, VirusFinder 2 runs the 4 steps sequentially on the input data and the virus identified in step (2) is used in steps (3) and (4) as virus reference genome. If the sequence of the virus being examined is known and provided as an input to VirusFinder 2, however, step (2) will be skipped to save time. User can also skip step (3) or (4) by specifying relevant variables in a configuration file (see subsection 3.1).

VirusFinder 2 also implemented our recently developed algorithm, Virus intEgration site detection through Reference SEquence customization (VERSE) (see right half of Figure 1). The idea behind VERSE is to improve detection through creating personalized reference genomes. Using 19 human tumors and cancer cell lines as test data, we found VERSE substantially enhanced the sensitivity of virus integration site detection (see section Citation, page 17). The input parameter of VERSE is described in subsection 3.1; its output is explained in subsection 5.1.

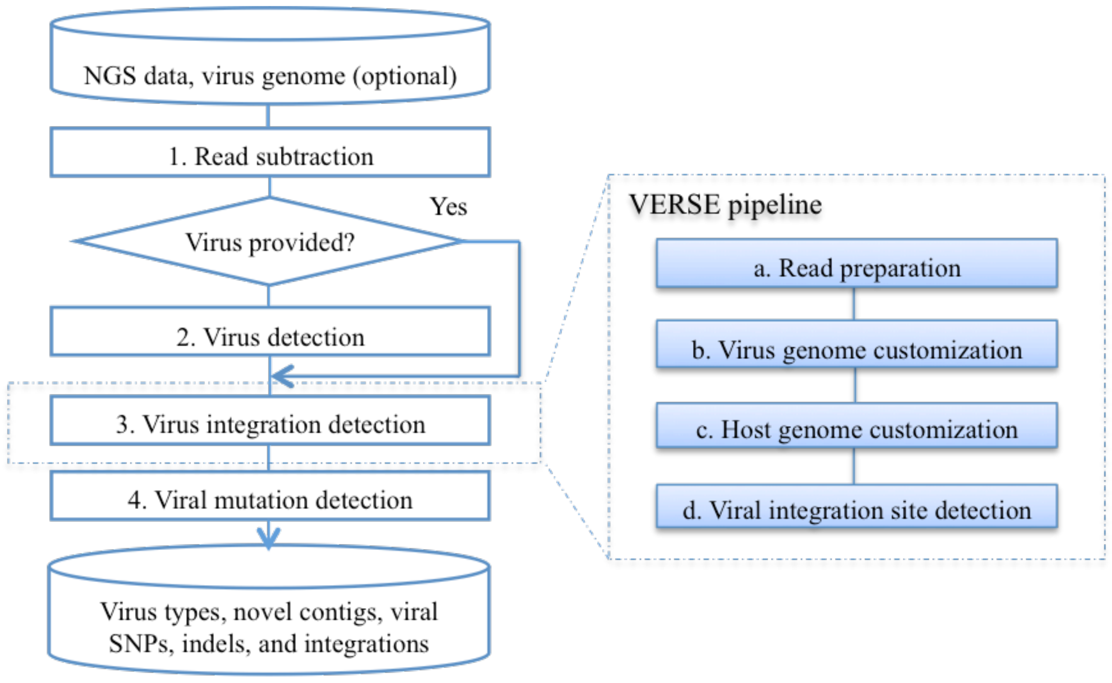
NGS data, virus genome (optional)
1. Read subtraction
Virus provided?
Yes
2. Virus detection
3. Virus integration detection
4. Viral mutation detection
Virus types, novel contigs, viral SNPs, indels, and integrations
VERSE pipeline
a. Read preparation
b. Virus genome customization
c. Host genome customization
d. Viral integration site detection

Figure 1: Overview of VirusFinder 2

1.1 Availability

VirusFinder 2 is available at http://bioinfo.mc.vanderbilt.edu/VirusFinder/.

1.2 Requirements

VirusFinder 2 requires: Linux, Bash shell, Java 1.6, and Perl 5.

1.3 Main features

A year ago, we publish VirusFinder 1.0 to detect viruses and their integration sites in paired-end NGS data (Wang *et al.*, 2013). We have since extended its functionalities and improved its accuracy over the past year based on numerous feedbacks from users. The new version, VirusFinder 2 not only incorporated all changes made to VirusFinder 1.x during the past year, but also provides new functionalities that help provide comprehensive characterization of intra-host viruses. The following Table summarizes main features of VirusFinder 2, in comparison with the earlier VirusFinder 1.0.

Table 1: Comparing VirusFinder 2 with VirusFinder 1.0

Category	Function	VirusFinder 1.0	VirusFinder 2
Input data	Single end		√
	Paired end	√	√
Virus detection	Report viruses	√	√
	Report novel contigs		√
	Report virus co-infections		√
Virus integration detection	Output integration sites	√	√
	Output consensus virus sequence		√
Viral mutation detection	Rare variants in virus genome		√

2. INSTALLATION

2.1 Source code

The four steps of VirusFinder 2 were implemented into four Perl scripts that can be run separately. For new users or users without programming skills, a wrapper script, VirusFinder.pl, is provided, as in previous VirusFinder 1.x, to allow user to complete all functionality using one simple command.

The following Table 2 provides a brief description of all the Perl scripts in VirusFinder 2. The script VirusFinder.pl prepares input data for each step of the pipeline and processes their outputs after they terminate. Scripts detect_virus.pl, detect_integration.pl, and detect_mutation.pl correspond to the steps (2), (3), and (4) of the pipeline, respectively. For the latest update of VirusFinder 2, please check our website at http://bioinfo.mc.vanderbilt.edu/VirusFinder/.

Table 2: Source code of VirusFinder

Script	Description
VirusFinder.pl	The interface of VirusFinder. It runs other Perl scripts to do actual work.
preprocess.pl	Subtracting host-derived reads and preparing input data for other script files.
detect_virus.pl	Detecting the existence of viruses in the data. If a virus sequence is provided to VirusFinder.pl, this script will be skipped by VirusFinder.pl.
detect_integration.pl	Detecting the integration sites of viruses in host genomes.

detect_mutation.pl	Identifying variants, i.e. SNPs and indels, in the virus genome.
Mosaic.pm	A Perl module consisting of subroutines shared by other Perl scripts.
sys_check.pl	An auxiliary script to check Java and Perl modules required by VirusFinder.

These Perl scripts need to be in the same directory in order to work properly. When executed with no argument or with the argument "-h", they will print detailed instructions for running them.

2.2 Third-party tools

Table 3 lists the third-party tools required by VirusFinder. Users have to set the directory of SAMtools in the environment variable PATH. For other tools, except BLAT, iCORN, CREST, and GATK, users should specify their full directories in the configuration file (see subsection 3.1 for a detailed description of the configuration file).

Table 3: Third party tools used in VirusFinder 2

Tool	Version	URL	Description
BLAST+	2.2.26+	ftp://ftp.ncbi.nlm.nih.gov/blast/executables/blast+/	Alignment tool
BLAT[1]	V.34	http://genome.ucsc.edu/cgi-bin/hgBlat	Alignment tool
Bowtie2	2	http://bowtie-bio.sourceforge.net/bowtie2/	Alignment tool
BWA	0.6.1	http://bio-bwa.sourceforge.net/	Alignment tool
iCORN[1]	0.97	http://icorn.sourceforge.net/	Reference correction tool
CREST[1]	1.1	http://www.stjuderesearch.org/site/lab/zhang	Structural variant calling
GATK[1]	2.4-9	http://www.broadinstitute.org/gatk/	Realignment tool
SAMtools	0.1.18	http://samtools.sourceforge.net/	Alignment processing tool
SVDetect	r0.8	http://svdetect.sourceforge.net/Site/	Structural variant calling
Trinity	2012-06-08	http://trinityrnaseq.sourceforge.net/	De novo assembly tool

[1]BLAT, GATK, and iCORN were already included in VirusFinder 2 under the directory *bin* and *icorn*, respectively. An in-house version of CREST as well as the external tools it requires, e.g. cap3, was provided in the directory *bin*. So user doesn't need to install them or specify their paths in the configuration file.

VirusFinder also needs several Perl modules, e.g. threads.pm for multithreading. The script sys_check.pl in Table 1 can help you identify the Perl modules that need to be installed in your system.

2.3 Reference genome

2.3.1 Host reference genome

The reference genome of the host of the virus is required for VirusFinder. Here, we use human reference genome, UCSC hg19 (http://hgdownload.cse.ucsc.edu/downloads.html#human), as an example to show how it works. The FASTA file hg19.fa needs to be indexed. The Bowtie2 and BLAST+ indices can be created using the following command lines.

```
bowtie2-build hg19.fa  hg19
makeblastdb -in hg19.fa -dbtype nucl -out hg19
```

2.3.2 Virus database (DB)

We used the same virus database (DB), virus.fa, as the one included with the RINS package (Bhaduri *et al.*, 2012, http://khavarilab.stanford.edu/resources.html). This virus DB contains viruses of all known classes (32,102 in total) (Bhaduri *et al.*, 2012). User can replace virus.fa with an alternative virus database, Genome Information Broker for Viruses (GIB-V, http://gib-v.genes.nig.ac.jp/) or NCBI viral gene annotation (ftp://ftp.ncbi.nih.gov/refseq/release/viral/). VirusFinder requires the BLAST+ index of the virus DB, which can be created using the following command:

```
makeblastdb -in virus.fa -dbtype nucl -out virus
```

The human reference genome and virus DB must be specified in the configuration file (see section 3.1).

3. GENERAL USAGE

3.1 Configuration file

The configuration file allows users to specify the full paths to their sequencing data and the third-party tools required by VirusFinder 2. It also allows users to adjust detection sensitivity by modifying the variables defined in the file. An example configuration file is provided in Figure 2, which shows that different input parameters are defined as "variable=value" pairs.

The first three variables in the configuration file specify the paths to the NGS data to be analyzed. The input of VirusFinder 2 can either be raw sequencing reads (in FASTQ format) or an alignment file (in BAM format). FASTQ files are preferred inputs. If user only specifies one FASTQ file using the variable "fastq1", VirusFinder 2 will treat the input as single end data; otherwise paired-end, if both "fastq1" and "fastq2" are provided.

By default, VirusFinder 2 runs all 4 steps of its pipeline sequentially. VirusFinder 2 will skip step (3), virus integration detection, if 'no' is assigned to variable "detect_integration". User can also skip step (4), viral mutation detection, by specifying "detect_mutation = no" in the configuration file.

A group of variables at the bottom of the configuration file enable users to adjust the sensitivity of virus detection. Specifically, the variable "min_contig_length" defines a contig length cutoff for the assembler Trinity. The default value 300 of "min_contig_length" works with most NGS data. However, if Trinity fails to assemble contigs of required length, e.g. 300, users can try to set smaller value to "min_contig_length" so as to make virus detection work. One new feature of VirusFinder 2 is that it can automatically adjust "min_contig_length" if Trinity fails to generate contigs of expected length.

For the rest of the variables, all are mandatory except "mailto", "detection_mode", "flank_region_size", and "sensitivity_level". If "detection_mode=sensitive", VirusFinder 2 will run the VERSE algorithm to detect virus integration sites. VERSE is more accurate than our previous algorithm implemented in VirusFinder 1.x. The variables "flank_region_size", and "sensitivity_level" allow users to tune the sensitivity of the VERSE algorithm (see Table 4 for detailed description). These three variables, i.e. "detection_mode", "flank_region_size", and "sensitivity_level", only apply to paired-end data. VirusFinder 2 will ignore them when working on single-end data. If not specified, VirusFinder 2 will use their default values.

```
############################################
## Input data can be:  (a) an alignment file (in BAM format); or (b) FASTQ file(s) (preferred) – for
##          single end data, "fastq1" is required; for paired-end, both "fastq1" and "fastq2" are needed.
############################################
alignment_file  = /scratch/kingw/virusFinder/simulation/simulation..bam
#fastq1            = /scratch/kingw/virusFinder/simulation/seq_1.fastq.gz
#fastq2            = /scratch/kingw/virusFinder/simulation/seq_2.fastq.gz

detect_integration = yes  # if 'no' is provided, step (3), virus integration detection, will be skipped
detect_mutation    = yes  # if 'no' is provided, step (4), viral mutation detection, will be skipped
mailto                 = qingguo.wang@vanderbilt.edu
thread_no            = 8     #the number of threads for parallel computing

############################################
## The full paths to the following third-party tools are required by VirusFinder:
############################################
blastn_bin          = /scratch/kingw/bin/ncbi-blast-2.2.26+/bin/blastn
bowtie_bin          = /scratch/kingw/bin/bowtie2-2.0.2/bowtie2
bwa_bin             = /scratch/kingw/bin/bwa-0.6.1/bwa
trinity_script      = /scratch/kingw/bin/trinityrnaseq_r2012-06-08/Trinity.pl
SVDetect_dir       = /scratch/kingw/bin/SVDetect_r0.8

############################################
## Reference files (indexed for Bowtie2 and BLAST)
############################################
virus_database           = /scratch/kingw/virusFinder/ref/virus.fa
bowtie_index_human = /scratch/kingw/virusFinder/ref/hg19
blastn_index_human  = /scratch/kingw/virusFinder/ref/hg19
blastn_index_virus    = /scratch/kingw/virusFinder/ref/virus

############################################
## Parameters of virus insertion detection (VERSE algorithm). They are ignored for single-end data
############################################
detection_mode     = sensitive    #Possible values: {normal, sensitive}; default value: normal.
                                  #If not specified, VirusFinder runs in normal detection mode.
flank_region_size  = 4000         #Suggested values: >2000; default: 4000; if detection_mode =
                                  #normal, it (and 'sensitivity_level' below) will be ignored.
sensitivity_level    = 1          #Suggested values: 1~6; default value: 1; greater value means higher
                                  #sensitivity, and accordingly more computation time.

############################################
## Parameters of virus detection. Smaller "min_contig_length", higher sensitivity
############################################
min_contig_length  = 300
blastn_evalue_thrd  = 0.05
similarity_thrd        = 0.8
chop_read_length    = 25
minIdentity            = 80
```

Figure 2: An example configuration file.

Table 4: Three modifiable variables to adjust the sensitivity of virus integration site detection

Variable	Description	Default value
detection_mode	The detection mode of VirusFinder. Possible values of this variable are: {normal, sensitive}. If not specified, VirusFinder runs in normal mode. In the normal mode, the two variables below will be ignored by VirusFinder.	normal
flank_region_size	VirusFinder examines both the flanking upstream region and flanking downstream region of the potential virus integration site-harboring regions. The larger size of the flanking regions examined by VirusFinder, the less likely VirusFinder misses virus insertion sites therein. This variable as well as the variable below only applies to the sensitive mode.	4000
sensitivity_level	VirusFinder improves virus integration site detection through iterative reference customization. This variable tells VirusFinder the number of customization iterations to run. Suggested values: 1~6. Larger value implies higher detection sensitivity.	1

3.2 Software interface VirusFinder.pl

VirusFinder.pl prints the following help information if run with no argument or with the argument "-h".

```
Program: VirusFinder, a tool for characterizing intra-host viruses through NGS data.
Version: 2 (06/19/2014)

Usage: VirusFinder.pl -c <configuration file> [options]

Options:
  -h, --help        Displays this information
  -c, --config      Configuration file <required>
  -v, --virus       The sequence file (in fasta format) of the virus. Not required
  -o, --output      The directory to store software output, default is current working directory
  -m, --markdup     Mark duplicate reads. Accepted inputs: y[es], n[o]. Default value is y[es].
                    Duplicate reads will not be used for variant calling. For ultra-deep amplicon
                    sequencing data, 'no' should be used for this argument.
```

Figure 3: Help information of VirusFinder.

A configuration file is required by VirusFinder as a mandatory input. Besides the configuration file, user can provide the virus being examined to VirusFinder if the virus infecting the host is already known. The option "-v" is not required, as VirusFinder has the capability to determine correct virus type from the NGS data. However, because virus detection is time-consuming, the offer of virus sequence to VirusFinder can bring significant timesaving. All intermediate and final results are stored in the directory specified by the option '-o'. If not provided, current working directory will be used. The last option "-m" applies to step (4) of the pipeline. Its meaning is self-explanatory and, thus, not elaborated here.

Figure 3 shows it is easy to run VirusFinder. The following command, though simple, should apply to most NGS projects (suppose "/home/kingw/project1" is a directory in which user wants to store results).

```
perl VirusFinder.pl -c config.txt -o /home/kingw/project1
```

3.3 Issue on virus co-infection

Virus integration detection and viral mutation detection are designed to deal with one virus at a time. For a sample co-infected with multiple viruses, e.g. HIV and EBV, although VirusFinder is able to identify all the viruses in the sample in its step (2), its automatic pipeline only chooses one, e.g. HIV, to analyze in steps (3) and (4). VirusFinder 2 provides flexible command line interface to assist users to characterize the integration sites/mutations of multiple viruses. For instance, to analyze the intra-host virus EBV, user can run the following two commands specifically,

```
perl detect_integration.pl -c config.txt -o /home/kingw/project1/step3-EBV -v gi_82503188
perl detect_mutation.pl    -c config.txt -o /home/kingw/project1/step4-EBV -v gi_82503188
```

Here, suppose "step3-EBV" and "step4-EBV" are two folders under /home/kingw/project1/ where user wants to store results. "gi_82503188" is the prefix of the ID of the EBV virus in the virus database (DB). For detailed descriptions of the scripts detect_integration.pl and detect_mutation.pl, please read the section below.

4. SPECIFIC USAGE

The previous section aims to help users quickly learn how to run VirusFinder 2. In this section, we describe in detail the four Perl scripts, preprocess.pl, detect_virus.pl, detect_integration.pl, and detect_mutation.pl, to prepare users for more complex applications. Most of the content below will be displayed on screen if the corresponding script is run with no argument or with the argument "-h".

4.1 Read subtraction

Table 5: Input and output of the script preprocess.pl

(a) Input

Argument	Description
-h, --help	Displays help information.
-c, --config	Configuration file <required>.
-o, --output	Full path of a directory to store results, default is current working directory.

(b) Output

File	Description
unmapped.1.fa	Reads unaligned to the host genome.
unmapped.2.fa	Reads unaligned to the host genome (only available for paired-end data).
unmapped.1.fq	Reads unaligned to the host genome. The reads in this file is the same as those in unmapped.1.fa.
unmapped.2.fq	Reads unaligned to the host genome (only available for paired-end data).

(c) Example

```
perl preprocess.pl -c config.txt
perl preprocess.pl -c config.txt -o /scratch/kingw/VirusFinder/simulation
```

4.2 Virus detection

Table 6: Input and output of the script detect_virus.pl

(a) Input

Argument	Description
-h, --help	Displays help information.
-c, --config	Configuration file <required>.
-o, --output	Full path of a directory to store results, default is current working directory.
--fa1	Sequencing reads (FASTA format). This file is mandatory unless there is a file unmapped.1.fa in the directory specified by the user. The file unmapped.1.fa can be created from FASTQ files using preprocess.pl, a script for subtracting reads mapped to the host genome.
--fa2	Sequencing reads (FASTA format). If this file is not provided and there isn't a file unmapped.2.fa in the directory specified by the user, the program will treat the input data as single end. Unmapped.2.fa can be created from FASTQ files using preprocess.pl, a script for subtracting reads mapped to the host genome.
-d,--dustcutoff	A DUST score is calculated for each contig. A contig is discarded if its DUST score is above the specified cutoff. The default value is 7.

(b) Output

File	Description
results-virus.txt	All likely viruses existing in the data.
results-virus-top1.fa	Sequence of the top-ranking virus candidate. It will be used for downstream viral mutation and viral integration site detection, if user does not provide virus sequence to VirusFinder.
results-virus-list.txt	A refined summary of viruses infecting the host.
results-novel-contig-1.fa	Novel contigs that do not align or partially align to the host genome.
results-novel-contig-2.fa	Novel contigs that do not align to the host genome.

(c) Example

```
perl detect_virus.pl -c config.txt
perl detect_virus.pl -c config.txt -o /scratch/kingw/VirusFinder/simulation
```

Both options --fa1 and --fa2 are not specified in the commands above. So before running the 1st command, user needs to place file unmapped.1.fa (and unmapped.2.fa for paired-end data) in current folder; for the 2nd command, the read file(s) need be in folder "/scratch/kingw/VirusFinder/simulation". Files unmapped.1.fa and unmapped.2.fa can be the outputs of the script preprocess.pl. When VirusFinder.pl is run, VirusFinder can automatically place these input files in correct place.

4.3 Virus integration detection

Table 7: Input and output of the script detect_integration.pl

(a) Input

Argument	Description
-h, --help	Displays help information.
-c, --config	Configuration file <required>.
-o, --output	Full path of a directory to store results, default is current working directory.
-v, --virus	The sequence file (in fasta format) of the virus. User can also provide the ID (or the prefix of the ID) of the virus in the virus database, e.g. gi_82503188. This argument is mandatory unless user provides a file results-virus-top1.fa under the directory specified by the user.
-m, --mode	The mode, either sensitive or normal, to run the program, default value is normal.
--fq1	Sequencing reads (FASTQ file). This file is mandatory unless there is a file unmapped.1.fq in the directory specified by the user. The file unmapped.1.fq can be created from FASTQ files using preprocess.pl, a script for subtracting reads mapped to the host genome.
--fq2	Sequencing reads (FASTQ format). If this file is not provided and there isn't a file unmapped.2.fq in the directory specified by the user. The program will treat the input data as single end. Unmapped.2.fq can be created from FASTQ files using preprocess.pl, a script for subtracting reads mapped to the host genome.

(b) Output

File	Description
results-virus-loci.txt	Identified virus integration sites (if present)
virus-corrected-seq.fa/ virus-consensus-seq.fa	The consensus sequence of the virus population in the host. VirusFinder 2 creates this file only for paired-end data when in sensitive detection mode

(c) Example

```
perl detect_integration.pl -c config.txt -o /scratch/kingw/VirusFinder/simulation
perl detect_integration.pl -c config.txt -v gi_82503188
perl detect_integration.pl -c config.txt -v /scratch/kingw/virus-ref/EBV.fa
```

The first command requires user to place a virus sequence file, results-virus-top1.fa, under the folder "/scratch/kingw/VirusFinder/simulation" beforehand. Otherwise, the program will complain. The file results-virus-top1.fa can be created using the script detect_virus.pl. In the second case, user specifies the ID of the EBV virus in the virus DB. The program will look up the virus DB to match the string "gi_82503188". The sequence of the first matched virus is used as reference for virus integration site detection. In the third example, a virus genome (FASTA format) is used directly as a reference.

4.4 Viral mutation detection

Table 8: Input and output of the script detect_mutation.pl

(a) Input

Argument	Description
-h, --help	Displays help information.
-c, --config	Configuration file <required>.
-o, --output	Full path of a directory to store results, default is current working directory.
-v, --virus	The sequence file (in fasta format) of the virus. User can also provide the ID (or the prefix of the ID) of the virus in the virus database, e.g. gi_82503188. This argument is mandatory unless user provides a file results-virus-top1.fa under the directory specified by the user.
--fq1	Sequencing reads are FASTQ file. This file is mandatory unless there is a file unmapped.1.fq in the directory specified by the user. The file unmapped.1.fq can be created from FASTQ files using preprocess.pl, a script for subtracting reads mapped to the host genome.
--fq2	Sequencing reads are FASTQ file. If this file is not provided and there isn't a file unmapped.2.fq in the directory specified by the user. The program will treat the input data as single end. Unmapped.2.fq can be created from FASTQ files using preprocess.pl, a script for subtracting reads mapped to the host genome.
-m, --markdup	Mark duplicate reads. Accepted inputs: y[es], n[o]. Default value is y[es]. Duplicate reads will not be used for variant calling. For ultra-deep amplicon sequencing data, 'no' should be used for this argument.

(b) Output

File	Description
mutations.vcf	Identified rare mutations, i.e. SNPs and indels, in the virus genome.
virus-consensus-variant.txt	Consensus variants in the virus genome.

(c) Example

```
perl detect_mutation.pl -c config.txt
perl detect_mutation.pl -c config.txt -v gi_82503188
```

As in previous examples, user needs to place a file, results-virus-top1.fa, under the current directory before running the first command. In the second case, the program uses the virus in the virus DB that matches the string "gi_82503188" as reference for read alignment and mutation calling.

5. OUTPUT

Under the working directory of VirusFinder, multiple files, e.g. 'virus.txt', 'virus-list.txt', 'contig.txt', 'integration-sites.txt', and 'viral-mutation.vcf', which contain the final results, will be created upon the termination of the method. These files as well as the intermediate files generated by VirusFinder are introduced in subsections below.

5.1 Intermediate results

VirusFinder keeps all intermediate results so that it does not have to restart the whole process from scratch if the server, to which it is assigned to run, fails. The drawback of this design is that users may have to manually delete the files that are improperly created by VirusFinder as a result of system failure.

Typically, each run of VirusFinder 2 creates four folders, 'step1', 'step2', 'step3', and 'step4' under the directory specified by users. Each of these four folders stores the intermediate files of the corresponding step of VirusFinder 2. For WGS data with very high coverage, e.g., 120×, the size of the intermediate files in the folder 'step1' can be close to 0.5TB. So we remind users to delete them after the analysis process terminates.

On single-end data, or when running in the normal mode on paired-end data, VirusFinder always creates two subfolders, 'crest' and 'SVDetect', under the directory 'step3', to store the results of CREST and SVDetect, respectively.

As aforementioned, on paired-end data and in the sensitive mode, VirusFinder 2 runs the VERSE algorithm to detect virus integration sites, which creates two subfolders, 'a-vfix' and 'b-align', in the directory 'step3' instead. The subfolder 'a-vfix' stores intermediate files of the consensus virus sequence, which is derived by correcting SNPs and indels in the reference virus genome, results-virus-top1.fa. 'b-align' consists of two subfolders, 'crest' and 'SVDetect', to store results of CREST and SVDetect, respectively. If VERSE does not find virus integration sites after the two sub-steps 'a-vfix' and 'b-align', it will decide either to terminate or run two additional steps, 'c-rfix' and 'd-align', based on the results thus far. The resulting subfolder 'c-rfix' stores the intermediate results of host genome customization. The subfolder 'd-align' stores the final detection results.

5.2 Output of virus detection

At the end of step (2), i.e., the virus detection step, at least three resultant files will be copied out of the subdirectory 'step2' to be placed in the working directory of VirusFinder. The three tab-delimited files, named 'virus.txt', 'virus-list.txt' and 'contig.txt' respectively, contain candidate viruses identified by VirusFinder and the corresponding contigs mapped to the virus sequences.

The candidate viruses in the file 'virus.txt' are sorted based on the alignment quality of the contigs that are mapped to them. The top-ranking virus candidate is in the first row of this file. Table 9 below shows the top three candidate virus sequences identified from our simulation data, which can be downloaded from our website (http://bioinfo.mc.vanderbilt.edu/VirusFinder/). All these three candidates indicate the existence of HPV-16 virus in the sample and HPV-16 is exactly what we inserted into this simulation data initially. One contig, comp4_c0_seq1, was mapped to all these three candidate sequences. The length of the contig and the number of reads fallen on it are also provided in the file 'virus.txt'.

It is worth mentioning that at the end of step (2), VirusFinder extracts the sequence of the top-ranking virus, i.e. 'gi_310698439_ref_NC_001526.2__Human_papillomavirus_type_16' in the case of our simulated data, from the virus DB and saves it as a file named 'results-virus-top1.fa' under the folder 'step2'. This file is used as a separate pseudo-chromosome 'chrVirus' to be concatenated with the reference host (e.g. human) chromosomes for the detection of virus integration sites in step (3). It is also used in step (4) for viral mutation detection.

Table 9: The top-ranking viruses identified from our simulation data

Virus name	Contig	Contig length (bp)	Mapped length /rate of contigs	# Reads fallen on contigs
gi_310698439_ref_NC_001526.2__Human_papillomavirus_type_16	comp4_c0_seq1	8253	7913/ 98.98	3916
gi_56463023_gb_AY686584.1__Human_papillomavirus_type_16_isolate_Qv17722E	comp4_c0_seq1	8253	7909/99.03	3916
gi_56462996_gb_AY686581.1__Human_papillomavirus_type_16_isolate_Qv15521E	comp4_c0_seq1	8253	7909/98.99	3916
……	……	……	……	……

As shown in Table 9, the file 'virus.txt' is very noisy, as it enumerates all likely viruses, to which each contig aligns. To facilitate the studies of virus co-infections, VirusFinder 2 creates a clean file 'virus-list.txt' to include only non-redundant viruses that have the best mapping with contigs. The following is a file created for the simulation data. There is only one row of data in this file. With the concise size of 'virus-list.txt', co-infection of multiple viruses (if present) can be easily discerned from it.

Of note, in Table 10, the column 'Identities' measures the similarity between a contig and a virus sequence. For example, the value 98.98 in the Table means 98.98% of the contig can be aligned to the virus genome gi_310698439_ref_NC_001526.2__Human_papillomavirus_type_16.

Table 10: The content of the file virus-list.txt created for our simulation data

Virus name	Contig	Contig length (bp)	Identities(%)	# Reads fallen on contigs
gi_310698439_ref_NC_001526.2__Human_papillomavirus_type_16	comp4_c0_seq1	8253	98.98	3916

Another file 'contig.txt' includes all high quality non-human contigs. Table 11 shows the first contig in the 'contig.txt' file created for our simulation data. VirusFinder 2 also output novel contigs (if present). The format of the novel contig files is the same as 'contig.txt' and hence will not be elaborated here.

Table 11: The first contig in the file contig.txt created for our simulation data

Contig name	# Reads	Virus	E-value	Bit score	Sequence
comp4_c0_seq1	3916	gi_56463023_gb_AY686584.1__Human_papillomavirus_type_16_isolate_Qv17722E	0	1.42E+04	AGGTTCTAG CAATTGTCGT GCCTCAG… …

5.3 Output of integration site detection

The file integration-sites.txt reports the positions of all detected virus insertion sites in the host genome. In our simulation data, we plugged one mutated copy of the reference HPV-16 virus at the position chr1:24020700 of the reference human genome. The following table provides the content of the integration-sites.txt file created by VirusFinder for this simulation data.

Table 12: The content of file integration-sites.txt created by VirusFinder for our simulation data

Chromosome 1	Position 1	Strand 1	Chromosome 2	Position 2	Strand 2	#Support reads (pair+softclip)	Confidence
chr1	24,020,709	+	chrVirus	1	+	13+15	high

The virus integration site in Table 12 involves two breakpoints, one at position 24,020,709 of the human chromosome 1 and another at position 1 of the HPV-16 virus sequence. The second to the last column indicates how many NGS reads support this detection. The word *pair* means the number of paired-end reads whose one end mapped to the human genome and another end aligned to the HPV-16 virus sequence. The word *softclip* means the number of reads that actually harbor the integration breakpoint within themselves. *Softclip* is the total soft-clipped reads that map to the human reference genome plus the total soft-clipped reads that aligned to the virus sequence.

The final column reports the confidence of VirusFinder in the actual position of virus-host integration. High confidence means there are sufficient soft-clipped reads to support the virus integration locus. Low confidence, however, indicates lack of soft-clipped reads for the accurate characterization of the locus.

It may be worth mentioning that VirusFinder's predictions of virus integration sites, especially the high confidence ones, are very close to the real positions. On most WGS samples, the virus integration loci predicted by VirusFinder are only several base pairs difference from the real ones. However, VirusFinder's ability to predict virus insertion is affected by sequencing coverage, read quality, read mapping, difference between the human genome and virus sequence under study, etc. This is the case not only to VirusFinder, but also true to other virus insertion-detecting tools. Low quality of any of these factors could result in the deterioration in detection performance.

5.4 Output of viral mutation detection

In step (4) of VirusFinder 2, the input virus reference genome, results-virus-top1.fa, is customized to fit the reads if the data is paired-end. By removing common viral SNPs and indels from the reference, a consensus virus reference genome, virus-consensus-seq.fa, is created for variant calling.

The detected SNPs and indels are saved in a file 'mutation.vcf'. When VirusFinder 2 terminates, this file is copied from the sub-folder 'step4' into the working directory of VirusFinder 2 (renamed as 'viral-mutation.vcf').

The file 'mutation.vcf' is in VCF format. Table 13 below shows an example 'mutation.vcf' (the first five columns). The header of the file, which provides detailed description of each field of the file, is not shown in Table 13.

Table 13: Part of an example mutation.vcf file

#CHROM	POS	ID	REF	ALT
chrVirus	3008	.	T	G
chrVirus	3270	.	G	A
chrVirus	5403	.	C	T
chrVirus	5742	.	ATCATG	A
chrVirus	6074	.	T	C

6. FAQ

6.1 Can VirusFinder deal with single-end sequencing data?

VirusFinder 2 is capable of analyzing single-end data. Previous VirusFinder 1.x versions, however, can only deal with paired-end data.

6.2 Can VirusFinder be applied to genomes of non-human species?

Yes, VirusFinder can detect viruses in arbitrary types of genomes. To run VirusFinder on a non-human genome, e.g. mouse_ref.fa (the suffix ".fa" required), you have to index the reference genome first,

```
bowtie2-build mouse_ref.fa  mouse_ref
makeblastdb -in mouse_ref.fa -dbtype nucl -out mouse_ref
```

Then, specify the location of the reference and the corresponding indices in the configuration file as follows:

```
bowtie_index_human = /scratch/kingw/VirusFinder/ref/mouse_ref
blastn_index_human  = /scratch/kingw/VirusFinder/ref/mouse_ref
```

The chromosome names of the non-human genome should not contain symbols like ':' and '_'.

6.3 The path of my BAM file is correct but I received this error while running VirusFinder: "Can't find BAM file...Step 1 terminated abnormally"?

Please double check if your configuration file is created on *Windows* or *Mac*. *Windows/Mac* machines have different line endings from *Unix/Linux*. Copy and paste across multiple platforms can cause this problem on *Linux.*

6.4 How to fix this error: "Step 1 terminated abnormally! Please delete intermediate files and try again"?

VirusFinder utilizes unmapped/partly mapped reads to detect viruses. The BAM files created by some aligners do not contain unmapped reads or is formatted in a way not recognized comfortably by samtools. If provided with such alignment files as input, VirusFinder will throw out this error.

If you encounter this error or are not sure whether your BAM file is appropriate for virus detection, we suggest using raw sequencing files (in FASTQ format) instead of a BAM file as input of VirusFinder. VirusFinder is very efficient. It should be able to terminate quickly even supplied with raw sequencing reads.

6.5 In step 2 of VirusFinder, blat found 0 match and Trinity crashed.

The old versions of VirusFinder, i.e. version 1.1 or older, need user to specify a path to blat V.34 in the configuration file. It will report this error if other blat version, e.g. V.35, is provided.
Since VirusFinder 1.2 (October 18, 2013), VirusFinder includes a blat V.34 in its release package. So an easy way to fix this error is to update your VirusFinder to a new version.

6.6 In step 2.3 of VirusFinder, I received this error: "Trinity did not output contigs. Please make sure Trinity works."

Two reasons could cause this error.

Firstly, the problem could be with the software Trinity. So please make sure Trinity is installed well before running VirusFinder. Trinity (version 2012-06-08) used in VirusFinder requires Java 1.6 (higher Java version may not work) and the Perl module threads.pm. Please make sure you have them in your system. The script sys_check.pl in VirusFinder can help check java version and the required Perl modules. Please make sure you do not receive failed report when running sys_check.pl.

Another reason causing this error could be that Trinity worked well but the preset contig length is too long. In this case, please try to set smaller value to "min_contig_length" in the configuration file.

In VirusFinder 2, we improved the code handling this error so that user can clearly differentiate the two conditions that result in this error.

6.7 In the file virus.txt, the top one entry is different from the virus I expect.

The virus at the top row of file 'virus.txt', a copy of file results-virus.txt under the directory step2, is used as reference for virus integration detection and viral mutation detection. If it is wrong virus type, VirusFinder can fail to detect virus integration sites later on.

If this happens, please modify results-virus.txt by deleting the first entry from it – suppose the one that follows is a correct virus type. Next, delete file results-virus-top1.fa, which is also under the directory step2. If directories step3 and step4 were already created by VirusFinder, then delete them. After that, rerun VirusFinder. VirusFinder will use the top entry in the updated file results-virus.txt to recreate the file results-virus-top1.fa, with which as reference to detect virus insertion sites and viral mutations.

6.8 "remove_tree" is not exported by the File::Path module.

The Perl module File::Path provides function remove_tree() since version 2.08. So this error indicates that an older version of File::Path is installed in user's server. To fix this error, user is encouraged to upgrade module File::Path to at least v2.08.

An alternative way is to modify the SVUtil.pm module, which is under the subdirectory 'bin' of VirusFinder, by replacing 'remove_tree' with 'rmtree', the legacy interface of remove_tree() with behavior and return value identical to remove_tree().

6.9 VirusFinder issued some warnings such as "SV filter starting.... FAILED".

This warning is not a software error. It is the output of the last step of CREST, a program run by VirusFinder. It means that low confidence detection did not pass the quality filter of CREST. Hence, it demonstrates that CREST has run successfully on your data.

6.10 The Perl module threads.pm is not installed in our server and I don't have root privilege. Is there anyway to run VirusFinder without multithreading?

The Perl module threads.pm is required by VirusFinder. You don't need root privilege to install it. You can install it under your local directory. After that, all you need to do to run VirusFinder is to inform Perl of the directory of threads.pm through option "-I",

```
perl -I  /path-of-your-threads.pm VirusFinder.pl -c config.txt
```

6.11 The file results-virus-loci.txt under the directory SVDetect is empty and I got an empty file integration-sites.txt. What else can I do to tune VirusFinder?

An empty *integration-sites.txt* probably means no virus integration event. To make sure VirusFinder did not miss real virus integration site, the first thing to check is the virus type that VirusFinder detected in step 2. If it is not right, please use correct virus reference to run detect_integration.pl again. Another thing to look into is the variables in the configuration file, e.g. flank_region_size and sensitivity_level.

Lastly, if you know a specific region in the host genome, into which the virus of your interest typically fuses, you can manually add a line into results-virus-loci.txt (see below - suppose the virus integrated into a locus within *chr9:121416995-121417300*). The file *results-virus-loci.txt* in the folder *SVDetect* is tab-delimited. It is used by CREST to narrow down search regions. The line below tells CREST to specifically check the region *chr9:121416995-121417300* for potential virus integration loci.

```
INTER	REVERSE_SENSE	-	chr9	121416995-121417300	-	chrVirus4776-5167
	8
```

6.12 Can VirusFinder be applied to SOLiD fastq files?

To analyze SOLiD fastq files, users have to covert color space reads into nucleotides, using tools such as solid2solexa (http://sourceforge.net/projects/solid2solexa/files/latest/download).

Solid2solexa truncates reads with illegal color code. It may also generate reads with illegal character '_' at the end of the nucleotide sequences. These problematic reads can cause software failure in downstream analysis. So before running VirusFinder, we suggest users to double-check its output carefully and filter out those problematic reads. After that, users should be able to run VirusFinder on their data.

7. REFERENCES

Wang Q, Jia P, Zhao Z (2013) VirusFinder: software for efficient and accurate detection of viruses and their integration sites in host genomes through next generation sequencing data. *PLoS ONE* 8(5):e64465.

Bhaduri A, Qu K, Lee CS, Ungewickell A, Khavari PA. (2012) Rapid identification of nonhuman sequences in high throughput sequencing data sets, *Bioinformatics*, 28:1174-1175.

Chen Y, Yao H, Thompson EJ, Tannir NM, Weinsten JN, Su X (2013) VirusSeq: software to identify viruses and their integration sites using next generation sequencing of human cancer tissue, *Bioinformatics*, 29(2):266-267.

Kostic AD, Ojesina AI, Pedamallu CS, Jung J, Verhaak RG, Getz G, Meyerson M. (2011) PathSeq: software to identify or discover microbes by deep sequencing of human tissue, *Nat Biotechnol*, 29:393-396.

Li JW, Wan R, Yu CS, Co NN, Wong N, Chan TF. (2013) ViralFusionSeq: accurately discover viral integration events and reconstruct fusion transcripts at single-base resolution, *Bioinformatics*, 29(5):649-651.

8. CITATION

Wang Q, Jia P, Zhao Z (2015) VERSE: a novel approach to detect virus integration in host genomes through reference genome customization. *Genome Medicine*, 7(1):2.